1.8 Nested Simple Recursion

1.8.1 Introduction. In this section we will investigate recursive definitions for which parameters may be arbitrarily substituted for, even with nested recursive applications. For instance:

$$f(0,y) = g(y)$$
$$f(x+1,y) = h\Big(x, f\big(x, \sigma[x,y,f(x,y)]\big), y\Big).$$

Such recursion is called nested simple recursion (see [1, 2]).

1.8.2 Notation. Let $\tau[f]$ be a term which may apply an n-ary function symbol f and $\vec{x}$ pairwise different n variables. We will use the *special lambda notation* $\tau[\dot{\lambda}\vec{x}.\rho[\vec{x}]]$, for the term obtained from τ by the replacement of all applications $f(\vec{\theta})$ in it by terms $\rho[\vec{\theta}]$.

1.8.3 Nested simple recursion. Let $\rho[\vec{y}]$ and $\tau[f;x,\vec{y}]$ are terms in which no other variables than the indicated ones are free. Suppose that ρ does not apply f. Consider the $(n{+}1)$-ary function f defined by

$$f(0,\vec{y}) = \rho[\vec{y}] \tag{1}$$
$$f(x+1,\vec{y}) = \tau[\dot{\lambda}x_1\vec{y}.f(x,\vec{y});x,\vec{y}]. \tag{2}$$

We say that f is defined by *nested simple recursion*. The definition can be viewed as a function operator which takes all auxiliary functions applied in the terms ρ,τ and yields the function f as a result.

The fact that p.r. functions are closed under simple nested recursion will be proved at the end of this section. The proof proceeds in stages. First, we prove the claim for the scheme with two recursive applications ($k = 2$) and one parameter ($n = 1$). This is proved in Thm. 1.8.11 by reducing the scheme to course of values recursion with parameter substitution. Next, we extend this result to the scheme with arbitrary number of recursive applications (Thm. 1.8.14). Finally, we prove the claim for the scheme with arbitrary number of parameters (Thm. 1.8.17).

We may assume that the function f is applied in τ at least once because otherwise there would be nothing to prove. In order to simplify our discussion we transform the equation (2) into equivalent one by 'unnesting' all recursive applications of f in the term τ:

$$\bigwedge_{i=1}^{k} f(x,\vec{\sigma}_i[x,\vec{y},\vec{z}_{i-1}]) = z_i \to f(x+1,\vec{y}) = \theta[x,\vec{z},\vec{y}]. \tag{3}$$

Here $\vec{z}_i$ abbreviates $z_1,\ldots,z_i$ and the terms $\sigma_1,\ldots,\sigma_k,\theta$ contain at most the indicated variables and do not apply f.

Nested Simple Recursion: Case $k = 2$ and $n = 1$

1.8.4 Introduction. In this subsection we will investigate the scheme of nested simple recursion with two different recursive applications ($k = 2$) and one parameter ($n = 1$). The admissibility of the scheme in the class of p.r. functions will be shown in Thm. 1.8.11.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following nested simple recursion:

$$f(0,y) = \rho[y] \tag{1}$$

$$f(x+1,y) = \theta\big[x, f\big(x,\sigma_1[x,y]\big), f\big(x,\sigma_2[x,y,f(x,\sigma_1[x,y])]\big), y\big] \tag{2}$$

from p.r. functions. We claim that f is also primitive recursive. We will prove this fact by reducing the scheme to previous recursive schemes.

1.8.5 The outline of the proof. We will introduce the function f as primitive recursive by arithmetization of its computation trees in which we use as computational rules the defining axioms 1.8.4(1)(2). The evaluation of the application $f(x,y)$ is recorded as a full binary tree of depth $x+1$ with labels consisting of all applications $f(x_i,y_i)$ which are needed to calculate the value $f(x,y)$.

Binary trees are coded as follows. The empty tree is coded by the number 0. A non-empty tree is coded by the number $\langle z,l,r\rangle$, where z is the label of its root node, and l and r are the codes of its left and right subtree, respectively. Note that if t is the code of a non-empty tree then the label of its root node is the first projection of t, i.e. the number $\pi_1(t)$.

We intend to introduce the function f as primitive recursive with the help of its *course of values function* $\overline{f}$. The function $\overline{f}(x,y)$ yields the computation tree for the application $f(x,y)$. This can be expressed more formally by the following properties of the course of values function:

$$\overline{f}(0,y) = \big\langle f(0,y),0,0\big\rangle$$

$$\overline{f}(x+1,y) = \Big\langle f(x+1,y),\overline{f}\big(x,\sigma_1[x,y]\big),\overline{f}\big(x,\sigma_2\big[x,y,f(x,\sigma_1[x,y])\big]\big)\Big\rangle.$$

We will show that the course of values function is primitive recursive. Hence the explicit definition $f(x,y) = \pi_1\overline{f}(x,y)$ will derive f as a p.r. function.

Note that the natural evaluation strategy for evaluating the application $f(x,y)$ corresponds to *postorder traversal* of the computation tree $\overline{f}(x,y)$. Indeed, consider the computation tree from Fig. 1.5. The course of values sequence of its labels

$$f(x_0,y_0), f(x_1,y_1), f(x_2,y_2), \ldots, f(x_j,y_j), \ldots, f(x_i,y_i), \ldots$$

consists of all applications which are needed to compute its root value. The parameter y_i of each application $f(x_i, y_i)$ depends only on those values $f(x_j, y_j)$ which are directly before it: $j < i$. We refer to the number i as the (index of) *position* of the node $f(x_i, y_i)$ in the computation tree. Note also that the order in which the course of values sequence is sorted corresponds to postorder traversal of the computation tree.

Let us now consider a finite sequence of full binary trees of depth $x + 1$

$$t_0, t_1, t_2, \ldots, t_i, \ldots, t_{2^{x+1} \dot{-} 1},$$

where each tree t_i satisfies the following condition w.r.t. the application $f(x, y)$: *every subtree of t_i at position $j < i$ is the computation tree for the application $f(x_j, y_j)$.* Such trees will be called *partial computation trees* for the application $f(x, y)$. Note that the last tree $t_{2^{x+1} \dot{-} 1}$ of the sequence is in fact a computation tree for $f(x, y)$.

This suggests the following method for building of the computation tree $\overline{f}(x, y)$. We start by creating a 'dummy' full binary tree t_0 of depth $x + 1$. Suppose that after $i < 2^{x+1}$-steps we get a partial computation tree t_i for $f(x, y)$. The tree is updated at position i by $f(x_i, y_i)$ whereby we obtain a new partial computation tree t_{i+1}. After 2^{x+1} steps we get the computation tree $\overline{f}(x, y)$ for $f(x, y)$.

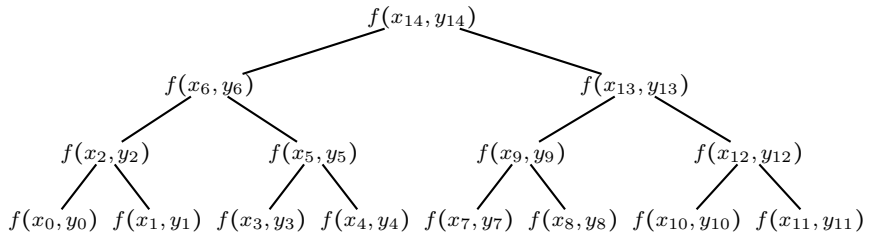

Fig. 1.5 Postorder traversal of a computation tree of depth 4

1.8.6 Full binary trees. The function $\mathit{Full}(n)$ creates a full binary tree of the depth n. The function is defined by primitive recursion as a p.r. function:

$$\begin{aligned} \mathit{Full}(0) &= 0 \\ \mathit{Full}(n+1) &= \langle 0, \mathit{Full}(n), \mathit{Full}(n) \rangle. \end{aligned}$$

1.8.7 Local node condition. The application $V(x, y, l, r)$ determines the correct value $f(x, y)$ from the subtrees l and r of a partial computation tree $\langle z, l, r \rangle$ for $f(x, y)$. The function is primitive recursive by the following explicit definition (with monadic discrimination):

$$V(0,y,l,r) = \rho[y]$$
$$V(x+1,y,l,r) = \theta[x,\pi_1(l),\pi_1(r),y]$$

1.8.8 Local update. The 4-ary function $U(t,i,x,y)$ updates the partial computation tree t for $f(x,y)$ at position i by $f(x_i,y_i)$. The function has the following basic properties

$$i < 2^x \dot{-} 1 \to U\big(\langle z,l,r\rangle,i,x,y\big) = \Big\langle z, U\big(l,i,x \dot{-} 1,\sigma_1[x \dot{-} 1,y]\big),r\Big\rangle \tag{1}$$

$$\begin{aligned} j < 2^x \dot{-} 1 \to U\big(\langle z,l,r\rangle, 2^x \dot{-} 1 + j,x,y\big) = \\ \Big\langle z,l,U\big(r,j,x \dot{-} 1,\sigma_2[x \dot{-} 1,y,\pi_1(l)]\big)\Big\rangle \end{aligned} \tag{2}$$

$$U\big(\langle z,l,r\rangle, 2^{x+1} \dot{-} 2,x,y\big) = \big\langle V(x,y,l,r),l,r\big\rangle. \tag{3}$$

Note that both 'recursive' applications of the function U on the right-hand side of the conditional equations (1) and (2) are applied to lesser arguments $l < \langle z,l,r\rangle$ and $r < \langle z,l,r\rangle$ than the one on the left. We will use this observation to find a course of values recursive definition of U as follows. The transformation of the specification properties into course of values derivation of U is based on the following simple properties of the projection functions:

$$\exists z \exists l \exists r\, t = \langle z,l,r\rangle \leftrightarrow \pi_2(t) \neq 0$$
$$t = \langle z,l,r\rangle \to z = \pi_1(t) \wedge l = \pi_1\pi_2(t) \wedge r = \pi_2^2(t)$$
$$\pi_1(t+1) \leq t \wedge \pi_1\pi_2(t+1) \leq t \wedge \pi_2^2(t+1) \leq t.$$

Now let ξ be the term

$$\begin{aligned} \xi[t,l_1,r_1,i,x,y] \equiv\ & D\Big(\pi_2(t), \\ & \quad D\Big(i{+}1 <_* 2^x, \\ & \qquad \big\langle \pi_1(t), l_1, \pi_2^2(t)\big\rangle, \\ & \qquad D\Big(i{+}2 <_* 2^{x+1}, \\ & \qquad\quad \big\langle \pi_1(t), \pi_1\pi_2(t), r_1\big\rangle, \\ & \qquad\quad \Big\langle V\big(x,y,\pi_1\pi_2(t),\pi_2^2(t)\big),\pi_1\pi_2(t),\pi_2^2(t)\Big\rangle\Big)\Big),0\Big). \end{aligned}$$

The function $U(t,i,x,y)$ is defined by course of values recursion on t with substitution in parameters as a p.r. function by

$$\begin{aligned} U(0,i,x,y) &= 0 \\ U(t+1,i,x,y) &= \xi\Big[t+1, U\big(\pi_1\pi_2(t+1), i, x \dot{-} 1, \sigma_1[x \dot{-} 1, y]\big), \\ &\qquad U\big(\pi_2^2(t+1), i \dot{-} (2^x \dot{-} 1), x \dot{-} 1, \sigma_2[x \dot{-} 1, y, \pi_1^2\pi_2(t+1)]\big), \\ &\qquad i, x, y\Big]. \end{aligned}$$

It is clear that the function U satisfies (1)-(3).

1.8.9 Global update. The 4-ary function $M_i(x,y,t)$ updates the partial computation tree t for the application $f(x,y)$ at each position $j < i$ by $f(x_j, y_j)$. The function is defined by primitive recursion on i as a p.r. function:

$$\begin{aligned} M_0(x,y,t) &= t \\ M_{i+1}(x,y,t) &= U(M_i(x,y,t), i, x, y). \end{aligned}$$

It has the following properties which will be needed in the sequel:

$$i+1 \le 2^{x+1} \to M_i\big(x+1, y, \langle z,l,r\rangle\big) = \Big\langle z, M_i\big(x, \sigma_1[x,y], l\big), r\Big\rangle \tag{1}$$

$$\begin{aligned} &i+1 \le 2^{x+1} \wedge M_{2^{x+1}\dot{-}1}\big(x, \sigma_1[x,y], l\big) = l_1 \to \\ &M_{2^{x+1}\dot{-}1+i}\big(x+1, y, \langle z,l,r\rangle\big) = \Big\langle z, l_1, M_i\big(x, \sigma_2[x,y,\pi_1(l_1)], r\big)\Big\rangle. \end{aligned} \tag{2}$$

Proof. (1): By induction on i. In the base case, clearly $0+1 \le 2^{x+1}$ and thus

$$M_0(x+1, y, \langle z,l,r\rangle) = \langle z,l,r\rangle = \big\langle z, M_0(x, \sigma_1[x,y], l), r\big\rangle.$$

In the induction step, if $(i+1)+1 \le 2^{x+1}$ then $i+1 \le 2^{x+1}$ and therefore

$$\begin{aligned} M_{i+1}(x+1,y,\langle z,l,r\rangle) &= U\big(M_i(x+1,y,\langle z,l,r\rangle), i, x+1, y\big) \overset{\text{IH}}{=} \\ &= U\big(\big\langle z, M_i(x,\sigma_1[x,y],l), r\big\rangle, i, x+1, y\big) \overset{1.8.8(1)}{=} \\ &= \Big\langle z, U\big(M_i(x,\sigma_1[x,y],l), i, x, \sigma_1[x,y]\big), r\Big\rangle = \big\langle z, M_{i+1}(x,\sigma_1[x,y],l), r\big\rangle. \end{aligned}$$

(2): By induction on i. In the base case suppose that $M_{2^{x+1}\dot{-}1}(x,\sigma_1[x,y],l) = l_1$. We clearly have $0+1 \le 2^{x+1}$ and thus

$$\begin{aligned} &M_{2^{x+1}\dot{-}1+0}(x+1,y,\langle z,l,r\rangle) = M_{2^{x+1}\dot{-}1}(x+1,y,\langle z,l,r\rangle) \overset{(1)}{=} \\ &= \big\langle z, M_{2^{x+1}\dot{-}1}(x,\sigma_1[x,y],l), r\big\rangle = \langle z,l_1,r\rangle = \Big\langle z, l_1, M_0\big(x, \sigma_2[x,y,\pi_1(l_1)], r\big)\Big\rangle. \end{aligned}$$

In the induction step, assume $(i+1)+1 \le 2^{x+1}$ and $M_{2^{x+1}\dot{-}1}(x,\sigma_1[x,y],l) = l_1$. Then $i+1 \le 2^{x+1}$ and we obtain

$$
\begin{aligned}
M_{2^{x+1}\dot{-}1+(i+1)}(x+1,y,\langle z,l,r\rangle) &= M_{2^{x+1}\dot{-}1+i+1}(x+1,y,\langle z,l,r\rangle) = \\
&= U\big(M_{2^{x+1}\dot{-}1+i}(x+1,y,\langle z,l,r\rangle),2^{x+1}\dot{-}1+i,x+1,y\big) \overset{\text{IH}}{=} \\
&= U\Big(\big\langle z,l_1,M_i\big(x,\sigma_2[x,y,\pi_1(l_1)],r\big)\big\rangle,2^{x+1}\dot{-}1+i,x+1,y\Big) \overset{1.8.8(2)}{=} \\
&= \Big\langle z,l_1,U\Big(M_i\big(x,\sigma_2[x,y,\pi_1(l_1)],r\big),i,x,\sigma_2[x,y,\pi_1(l_1)]\Big)\Big\rangle = \\
&= \big\langle z,l_1,M_{i+1}\big(x,\sigma_2[x,y,\pi_1(l_1)],r\big)\big\rangle. \qquad \square
\end{aligned}
$$

1.8.10 Course of values function. The binary function $\overline{f}(x,y)$ returns the computation tree for $f(x,y)$. The course of values function for f satisfies

$$\overline{f}(0,y) = \big\langle \rho[y],0,0\big\rangle \tag{1}$$

$$\begin{aligned}\overline{f}\big(x,\sigma_1[x,y]\big) = l \wedge \overline{f}\big(x,\sigma_2[x,y,\pi_1(l)]\big) = r \to \\ \overline{f}(x+1,y) = \big\langle \theta[x,\pi_1(l),\pi_1(r),y],l,r\big\rangle\end{aligned} \tag{2}$$

and it is defined explicitly as a p.r. function by

$$\overline{f}(x,y) = M_{2^{x+1}\dot{-}1}\big(x,y,\mathit{Full}(x+1)\big).$$

Proof. (1): It follows from

$$
\begin{aligned}
\overline{f}(0,y) &= M_{2^{0+1}\dot{-}1}\big(0,y,\mathit{Full}(0+1)\big) = M_1\big(0,y,\mathit{Full}(1)\big) = M_1\big(0,y,\langle 0,0,0\rangle\big) = \\
&= U\Big(M_0\big(0,y,\langle 0,0,0\rangle\big),0,0,y\Big) = U\big(\langle 0,0,0\rangle,0,0,y\big) \overset{1.8.8(3)}{=} \\
&= \big\langle V(0,y,0,0),0,0\big\rangle = \big\langle \rho[y],0,0\big\rangle.
\end{aligned}
$$

(2): Suppose that

$$
\begin{aligned}
\overline{f}\big(x,\sigma_1[x,y]\big) &= l \\
\overline{f}\big(x,\sigma_2[x,y,\pi_1(l)]\big) &= r.
\end{aligned}
$$

Then, by definition, we have

$$M_{2^{x+1}\dot{-}1}\big(x,\sigma_1[x,y],\mathit{Full}(x+1)\big) = l \tag{$\dagger_1$}$$

$$M_{2^{x+1}\dot{-}1}\big(x,\sigma_2[x,y,\pi_1(l)],\mathit{Full}(x+1)\big) = r \tag{$\dagger_2$}$$

and therefore

$$
\begin{aligned}
\overline{f}(x+1,y) &= M_{2^{x+1+1}\dot{-}1}\big(x+1,y,\mathit{Full}(x+1+1)\big) = \\
&= M_{2^{x+2}\dot{-}2+1}\big(x+1,y,\big\langle 0,\mathit{Full}(x+1),\mathit{Full}(x+1)\big\rangle\big) = \\
&= U\Big(M_{2^{x+2}\dot{-}2}\big(x+1,y,\big\langle 0,\mathit{Full}(x+1),\mathit{Full}(x+1)\big\rangle\big),2^{x+2}\dot{-}2,x+1,y\Big) =
\end{aligned}
$$

$$= U\Big(M_{2^{x+1}\dot{-}1+(2^{x+1}\dot{-}1)}\big(x+1, y, \langle 0, \mathit{Full}(x+1), \mathit{Full}(x+1)\rangle\big),$$
$$2^{x+2}\dot{-}2, x+1, y\Big) \overset{(\dagger_1),\ 1.8.9(2)}{=}$$
$$= U\Big(\big\langle 0, M_{2^{x+1}\dot{-}1}\big(x, \sigma_1[x,y], \mathit{Full}(x+1)\big),$$
$$M_{2^{x+1}\dot{-}1}\big(x, \sigma_2[x,y,\pi_1(l)], \mathit{Full}(x+1)\big)\big\rangle, 2^{x+2}\dot{-}2, x+1, y\Big) \overset{(\dagger_1),\ (\dagger_2)}{=}$$
$$= U\big(\langle 0,l,r\rangle, 2^{x+2}\dot{-}2, x+1, y\big) \overset{1.8.8(3)}{=} \big\langle V(x+1,y,l,r), l, r\big\rangle =$$
$$= \big\langle \theta[x, \pi_1(l), \pi_1(r), y], l, r\big\rangle. \qquad \square$$

1.8.11 Theorem *Primitive recursive functions are closed under nested simple recursion for the case $k = 2$ and $n = 1$.*

Proof. Let f be defined by nested simple recursion as in Par. 1.8.4 from p.r. functions. Let further $\overline{f}$ be its course of values function as in Par. 1.8.10. We claim that we have

$$f(x, \vec{y}) = \pi_1 \overline{f}(x, \vec{y}). \tag{1}$$

The function $\overline{f}$ is primitive recursive and so is f.

The property is proved by induction on x as $\forall y(1)$. In the base case take any y and we have

$$f(0,y) = \rho[y] = \pi_1\big\langle g(y), 0, 0\big\rangle \overset{1.8.10(1)}{=} \pi_1 \overline{f}(0,y).$$

In the induction step take any y and we obtain

$$f(x+1,y) = \theta\Big[x, f\big(x, \sigma_1[x,y]\big), f\Big(x, \sigma_2\big[x, y, f(x, \sigma_1[x,y])\big]\Big), y\Big] \overset{\text{IH}}{=}$$
$$= \theta\Big[x, f\big(x, \sigma_1[x,y]\big), \pi_1\overline{f}\Big(x, \sigma_2\big[x, y, f(x, \sigma_1[x,y])\big]\Big), y\Big] \overset{\text{IH}}{=}$$
$$= \theta\Big[x, \pi_1\overline{f}\big(x, \sigma_1[x,y]\big), \pi_1\overline{f}\Big(x, \sigma_2\big[x, y, \pi_1\overline{f}(x, \sigma_1[x,y])\big]\Big), y\Big] \overset{1.8.10(2)}{=}$$
$$= \pi_1 \overline{f}(x+1, y). \qquad \square$$

Nested Simple Recursion: Case n = 1

1.8.12 Introduction. In this subsection we will show that p.r. functions are closed under the scheme of nested simple recursion with one parameter. This will be proved in Thm. 1.8.14 by reducing the number of recursive applications. This leads eventually to nested simple recursion with one parameter and two recursive applications.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following nested simple recursion

$$f(0,y) = \rho[y] \tag{1}$$

$$\bigwedge_{i=1}^{k+1} f\big(x,\sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i \to f(x+1,y) = \theta[x,z_1,\dots,z_{k+1},y] \tag{2}$$

from p.r. functions. We claim that f is also primitive recursive.

1.8.13 Reduction of the number of recursive applications. We will reduce the above definition for $k \geq 2$ to a new one, where only k recursive applications are allowed. This new definition is for a binary function $\hat{f}(u,v)$ such that $\hat{f}(2x,y) = f(x,y)$ and it is of the form

$$\hat{f}(0,v) = \rho[v]$$

$$\bigwedge_{i=1}^{k} \hat{f}\big(u,\hat{\sigma}_i[u,v,\vec{w}_{i-1}]\big) = w_i \to \hat{f}(u+1,v) = \hat{\theta}[u,w_1,\dots,w_k,v]$$

for suitable terms $\hat{\theta}[u,\vec{w},v],\hat{\sigma}_1[u,v,\vec{w}_0],\dots,\hat{\sigma}_k[u,v,\vec{w}_{k-1}]$. Here $\vec{w}_i$ abbreviates $w_1,\dots,w_i$. Note that this is nested simple recursion on u with substitution in the parameter v with k recursive applications. We will take then the identity $f(x,y) = \hat{f}(2x,y)$ as an alternative, explicit definition of f.

The idea behind reduction of recursive applications is as follows. We would like to have $\hat{f}(2x,y) = f(x,y)$ and so it must be

$$\bigwedge_{i=1}^{k+1} \hat{f}\big(2x,\sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i \to \hat{f}\big(2(x+1),y\big) = \theta[x,z_1,\dots,z_{k+1},y]. \tag{1}$$

For the values of the form $\hat{f}(2x+1,v)$ we require

$$\bigwedge_{i=1}^{k} f\big(x,\sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i \to \hat{f}\big(2x+1,\langle 1,y\rangle\big) = \langle z_1,\dots,z_k\rangle$$

$$\hat{f}\big(2x+1,\langle 2,y,\vec{z}_k\rangle\big) = f\big(x,\sigma_{k+1}[x,y,\vec{z}_k]\big).$$

Note that the application $\hat{f}\big(2x+1,\langle 1,y\rangle\big)$ returns a number which codes k values $f\big(x,\sigma_1[x,y,\vec{z}_0]\big),\dots,f\big(x,\sigma_k[x,y,\vec{z}_{k-1}]\big)$ of the function f for some $\vec{z}$.

These informal arguments can be rewritten without mentioning the function f as follows:

$$\bigwedge_{i=1}^{k} \hat{f}\big(2x, \sigma_i[x, y, \vec{z}_{i-1}]\big) = z_i \to \hat{f}\big(2x+1, \langle 1, y\rangle\big) = \langle z_1, \ldots, z_k\rangle \tag{2}$$

$$\hat{f}\big(2x+1, \langle 2, y, \vec{z}_k\rangle\big) = \hat{f}\big(2x, \sigma_{k+1}[x, y, \vec{z}_k]\big) \tag{3}$$

$$\hat{f}\big(2x+2, y\big) = \theta\Big[x, \big[\hat{f}\big(2x+1, \langle 1, y\rangle\big)\big]_1^k, \ldots, \big[\hat{f}\big(2x+1, \langle 1, y\rangle\big)\big]_k^k, \hat{f}\Big(2x+1, \big\langle 2, y, \hat{f}(2x+1, \langle 1, y\rangle)\big\rangle\Big), y\Big]. \tag{4}$$

This means that the terms $\hat{\tau}, \hat{\sigma}_1, \ldots, \hat{\sigma}_k.$ satisfy the properties

$$\hat{\theta}[2x+1, z, z_{k+1}, \ldots, y] = \theta\Big[x, [z]_1^k, \ldots, [z]_k^k, z_{k+1}, y\Big] \tag{5}$$

$$\hat{\theta}\big[2x, z_1, \ldots, z_k, \langle 1, y\rangle\big] = \langle z_1, \ldots, z_k\rangle \tag{6}$$

$$\hat{\theta}\big[2x, z_{k+1}, \ldots, \langle 2, y, z\rangle\big] = z_{k+1} \tag{7}$$

and

$$\hat{\sigma}_1[2x+1, y] = \langle 1, y\rangle \tag{8}$$

$$\hat{\sigma}_2[2x+1, y, z] = \langle 2, y, z\rangle \tag{9}$$

$$\bigwedge_{i=1}^{k} \hat{\sigma}_i\big[2x, \langle 1, y\rangle, \vec{z}_{i-1}\big] = \sigma_i[x, y, \vec{z}_{i-1}] \tag{10}$$

$$\hat{\sigma}_1\big[2x, \langle 2, y, \vec{z}_k\rangle\big] = \sigma_{k+1}[x, y, \vec{z}_k]. \tag{11}$$

For that it is sufficient to set

$$\hat{\theta}[u, w_1, \ldots, w_k, v] \equiv D\Big(u \bmod 2, \theta\Big[u \div 2, [w_1]_1^k, \ldots, [w_1]_k^k, w_2, v\Big], D\big(\pi_1(v) =_* 1, \langle w_1, \ldots, w_k\rangle, w_1\big)\Big)$$

and

$$\hat{\sigma}_1[u, v] \equiv D\Big(u \bmod 2, \langle 1, v\rangle, D\Big(\pi_1(v) =_* 1, \sigma_1\big[u \div 2, \pi_2(v)\big], \sigma_{k+1}\Big[u \div 2, \pi_1\pi_2(v), \big[\pi_2^2(v)\big]_1^k, \ldots, \big[\pi_2^2(v)\big]_k^k\Big]\Big)\Big)$$

$$\hat{\sigma}_2[u,v,w_1] \equiv D\big(u \bmod 2, \langle 2,v,w_1\rangle, \sigma_2\,[u \div 2, \pi_2(v), w_1]\big)$$
$$\hat{\sigma}_i[u,v,\vec{w}_{i-1}] \equiv \sigma_i\,[u \div 2, \pi_2(v), \vec{w}_{i-1}] \quad \text{for } i = 3,\dots,k.$$

Proof. (5)-(11): Directly from definition. (2): Let us denote by $z_1,\dots,z_k$ the numbers such that $\hat{f}\big(2x, \sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i$ for every $i = 1,\dots,k$. We then have

$$\hat{f}\Big(2x, \hat{\sigma}_i\big[2x, \langle 1,y\rangle, \vec{z}_{i-1}\big]\Big) \overset{(10)}{=} \hat{f}\big(2x, \sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i \qquad (\dagger_1)$$

for every $i = 1,\dots,k$. From this we obtain

$$\hat{f}\big(2x+1, \langle 1,y\rangle\big) \overset{(\dagger_1)}{=} \hat{\theta}\big[2x, z_1,\dots,z_k, \langle 1,y\rangle\big] \overset{(6)}{=} \langle z_1,\dots,z_k\rangle.$$

(3): It follows from

$$\begin{aligned}\hat{f}\big(2x+1, \langle 2,y,\vec{z}_k\rangle\big) &= \hat{\theta}\Big[2x, \hat{f}\Big(2x, \hat{\sigma}_1\big[2x, \langle 2,y,\vec{z}_k\rangle\big]\Big), \dots, \langle 2,y\rangle\Big] \overset{(11)}{=} \\ &= \hat{\theta}\Big[2x, \hat{f}\big(2x, \sigma_{k+1}[x,y,\vec{z}_k]\big), \dots, \langle 2,y\rangle\Big] \overset{(7)}{=} \hat{f}\big(2x, \sigma_{k+1}[x,y,\vec{z}_k]\big).\end{aligned}$$

(4): It follows from

$$\begin{aligned}&\hat{f}(2x+2,y) = \hat{f}(2x+1+1,y) = \\ &= \hat{\theta}\Big[2x+1, \hat{f}\big(2x+1, \hat{\sigma}_1[2x+1,y]\big), \\ &\qquad \hat{f}\Big(2x+1, \hat{\sigma}_2\big[2x+1, y, \hat{f}(2x+1, \hat{\sigma}_1[2x+1,y])\big]\Big), \dots, y\Big] \overset{(8),(9)}{=} \\ &= \hat{\theta}\Big[2x+1, \hat{f}\big(2x+1, \langle 1,y\rangle\big), \hat{f}\Big(2x+1, \big\langle 2, y, \hat{f}(2x+1, \langle 1,y\rangle)\big\rangle\Big), \dots, y\Big] \overset{(5)}{=} \\ &= \theta\Big[x, \big[\hat{f}\big(2x+1, \langle 1,y\rangle\big)\big]_1^k, \dots, \big[\hat{f}\big(2x+1, \langle 1,y\rangle\big)\big]_k^k, \\ &\qquad \hat{f}\Big(2x+1, \big\langle 2, y, \hat{f}(2x+1, \langle 1,y\rangle)\big\rangle\Big), y\Big].\end{aligned}$$

We are now in position to prove (1). Suppose that

$$\bigwedge_{i=1}^{k+1} \hat{f}\big(2x, \sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i$$

Then by (2) and (3) we obtain

$$\hat{f}\big(2x+1, \langle 1,y\rangle\big) = \langle z_1,\dots,z_k\rangle \qquad (\dagger_2)$$
$$\hat{f}\big(2x+1, \langle 2,y,\vec{z}_k\rangle\big) = z_{k+1}. \qquad (\dagger_3)$$

We now have

$$\begin{aligned}\hat{f}\big(2(x+1),y\big) &= \hat{f}(2x+2,y) \overset{(4)}{=} \\ &= \theta\Big[x,\big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_1^k,\ldots,\big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_k^k, \\ &\qquad \hat{f}\Big(2x+1,\big\langle 2,y,\hat{f}(2x+1,\langle 1,y\rangle)\big\rangle\Big),y\Big] \overset{(\dagger_2),(\dagger_3)}{=} \\ &= \theta\Big[x,[\langle z_1,\ldots,z_k\rangle]_1^k,\ldots,[\langle z_1,\ldots,z_k\rangle]_k^k,z_{k+1},y\Big] = \\ &= \theta[x,z_1,\ldots,z_k,z_{k+1},y]. \end{aligned}$$ □

1.8.14 Theorem *Primitive recursive functions are closed under nested simple recursion for the case $n = 1$.*

Proof. The claim is proved by (meta-)induction on the number k of recursive applications in the defining axiom 1.8.12(2). The case $k = 0$ is in fact explicit definition with monadic discrimination on the first argument and it follows from Thm. 1.2.5. The cases $k = 1$ or $k = 2$ follow from Thm. 1.8.11. So suppose that the claim holds for the case $k \geq 2$. We will prove that the claim holds also for the case $k + 1$.

Let f be defined by nested simple recursion as in Par. 1.8.12 from p.r. functions. Let further $\hat{f}$ be the function from Par. 1.8.13. We claim that

$$f(x,y) = \hat{f}(2x,y). \tag{1}$$

The auxiliary function $\hat{f}$ is primitive recursive and so is f.

The property is proved by induction on x as $\forall y(1)$. In the base case take any y and we have

$$f(0,y) = \rho[y] = \hat{f}(0,y) = \hat{f}(2\times 0,y).$$

In the induction step take any y and let us denote by $z_1,\ldots,z_{k+1}$ the numbers such that

$$\bigwedge_{i=1}^{k+1} f\big(x,\sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i$$

By $(k{+}1)$ applications of IH we obtain

$$\bigwedge_{i=1}^{k+1} \hat{f}\big(2x,\sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i.$$

We then have

$$f(x+1,y) = \theta[x,z_1,\ldots,z_{k+1},y] \overset{1.8.13(1)}{=} \hat{f}\big(2(x+1),y\big).$$ □

Nested Simple Recursion

1.8.15 Introduction. In this subsection we will show that p.r. functions are closed under the scheme of nested simple recursion with arbitrary number of parameters. This will be proved in Thm. 1.8.17 by reducing it to nested simple recursion with one parameter.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following nested simple recursion

$$f(0,\vec{y}) = \rho[\vec{y}] \tag{1}$$

$$\bigwedge_{i=1}^{k} f\big(x,\vec{\sigma}_i[x,\vec{y},\vec{z}_{i-1}]\big) = z_i \to f(x+1,\vec{y}) = \theta[x,\vec{z},\vec{y}] \tag{2}$$

from p.r. functions. We claim that f is also primitive recursive.

Below we will consider the case when the definition has at least two parameters, i.e. $n \geq 2$. The case $n = 0$ is in fact parameterless primitive recursion for which the claim has been already proved in Thm. 1.2.5. The case with one parameter ($n = 1$) follows from Thm. 1.8.14.

1.8.16 Contraction of parameters. We will reduce the above scheme, where $n \geq 2$, to a new one for a binary function $\langle f\rangle(x,y)$ so that

$$\langle f\rangle(x,y) = f\left(x,[y]_1^n,\ldots,[y]_n^n\right).$$

The n parameters $\vec{y} \equiv y_1,\ldots,y_n$ are replaced by a single parameter y. We will call the number $y = \langle\vec{y}\rangle \equiv \langle y_1,\ldots,y_n\rangle$ the *contraction* of the numbers $\vec{y}$.

The *contraction* function $\langle f\rangle(x,y)$ is defined by nested simple recursion on x with one parameter y as a p.r. function by

$$\langle f\rangle(0,y) = \rho\left[[y]_1^n,\ldots,[y]_n^n\right] \tag{1}$$

$$\begin{aligned}\bigwedge_{i=1}^{k}\Big\langle f\Big\rangle\Big(x,\Big\langle\vec{\sigma}_i\big[x,[y]_1^n,\ldots,[y]_n^n,\vec{z}_{i-1}\big]\Big\rangle\Big) = z_i \to \\ \big\langle f\big\rangle\big(x+1,[y]_1^n,\ldots,[y]_n^n\big) = \theta\left[x,\vec{z},[y]_1^n,\ldots,[y]_n^n\right].\end{aligned} \tag{2}$$

1.8.17 Theorem *Primitive recursive functions are closed under nested simple recursion.*

Proof. Let f be defined by nested simple recursion as in Par. 1.8.15 from p.r. functions, where the number of parameters is at least two ($n \geq 2$).[2] Let further $\langle f\rangle$ be the contraction function of f from Par. 1.8.16. We claim that

$$f(x,\vec{y}) = \langle f\rangle\big(x,\langle\vec{y}\rangle\big). \tag{1}$$

[2] The cases $n = 0$ or $n = 1$ follow from Thm. 1.2.5 or Thm. 1.8.12, respectively.

The auxiliary function $\langle f \rangle$ is primitive recursive and so is f.

The property is proved by induction on x as $\forall \vec{y}(1)$. In the base case take any $\vec{y}$ and we have

$$f(0, \vec{y}) = \rho[\vec{y}] = \rho\left[[\langle \vec{y} \rangle]_1^n, \ldots, [\langle \vec{y} \rangle]_n^n\right] = \langle f \rangle\big(0, \langle \vec{y} \rangle\big).$$

In the induction step take any $\vec{y}$ and let us denote by $\vec{z}$ the numbers such that the following holds

$$\bigwedge_{i=1}^{k} f\big(x, \vec{\sigma}_i[x, \vec{y}, \vec{z}_{i-1}]\big) = z_i.$$

By k applications of IH we obtain

$$\bigwedge_{i=1}^{k} \langle f \rangle\Big(x, \big\langle \vec{\sigma}_i[x, \vec{y}, \vec{z}_{i-1}] \big\rangle\Big) = z_i$$

and thus we have

$$\bigwedge_{i=1}^{k} \langle f \rangle\Big(x, \big\langle \vec{\sigma}_i\big[x, [\langle \vec{y} \rangle]_1^n, \ldots, [\langle \vec{y} \rangle]_n^n, \vec{z}_{i-1}\big] \big\rangle\Big) = z_i.$$

From this we obtain

$$f(x+1, \vec{y}) = \theta[x, \vec{z}, \vec{y}] = \theta\left[x, \vec{z}, [\langle \vec{y} \rangle]_1^n, \ldots, [\langle \vec{y} \rangle]_n^n\right] = \langle f \rangle\big(x+1, \langle \vec{y} \rangle\big). \qquad \square$$